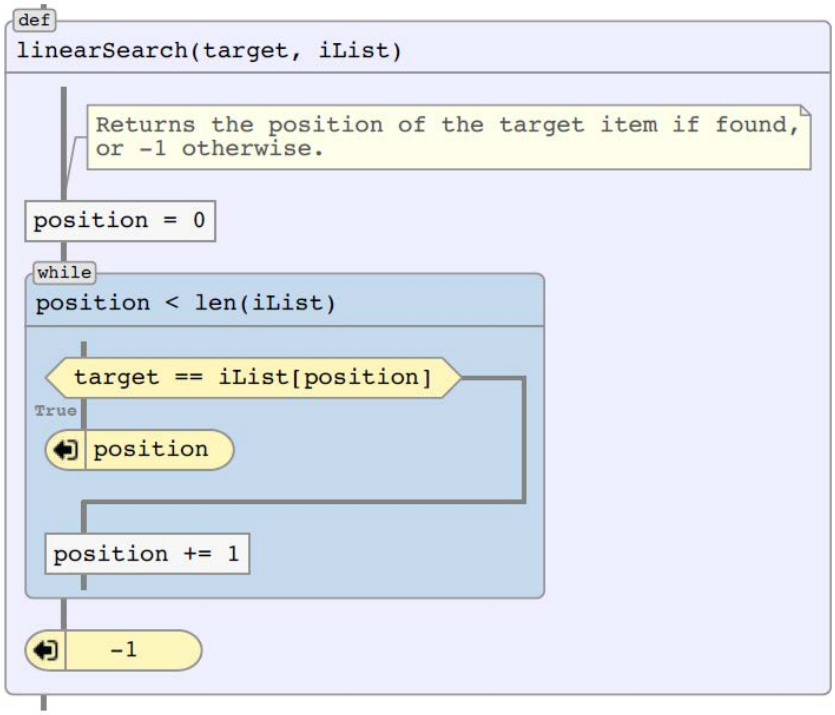
CIND830
Python Programming for Data Science
Lab 7 - Questions

1 Search Algorithms

The following algorithm searches a given list sequentially to find the position of a given item. Notably, the performance of that algorithm depends on the location of the target item.



a. Redesign the linear algorithm to start searching from the end of the list rather than from its beginning.

b. Write the code of the algorithm in **1a** then verify its correctness by running the following commands.

```
firstLinearSearch = linearSearch(4, [4, 5, 8, 4, 9, 3, 6, 2])
print(firstLinearSearch)

secondLinearSearch = linearSearch(9, [4, 5, 8, 4, 9, 3, 6, 2])
print(secondLinearSearch)

thirdLinearSearch = linearSearch(2, [4, 5, 8, 4, 9, 3, 6, 2])
print(thirdLinearSearch)
```

c. Compare the results obtained in question **1b** with those of the original linear search that starts the searching process from the beginning of the list.

d. Which of the tests in question **1b** represents the best case scenario in terms of the `Big O Notation`? Justify your answer.

e. Would the best-case scenario for the modified linear search algorithm be the best scenario for the original one?

2 Sort Algorithms

The following code defines an algorithm that sorts a list in ascending order.

```
def selectionSort(lyst):
    i = 0
    while i < len(lyst) - 1:          # Do n - 1 searches
        minIndex = i                  # for the smallest
        j = i + 1
        while j < len(lyst):          # Start another search
            if lyst[minIndex] > lyst[j]:
                minIndex = j
            j += 1
        if minIndex != i:             # Swap if needed
            lyst[minIndex], lyst[i] = lyst[i], lyst[minIndex]
        i += 1
    return(lyst)
```

a. Verify the correctness of that algorithm by running the following code and displaying the results.

```
selectionSort = selectionSort([5, 6, 8, 3, 2, 1, 9, 4])
print(selectionSort)
# The output should be [1, 2, 3, 4, 5, 6, 8, 9]
```

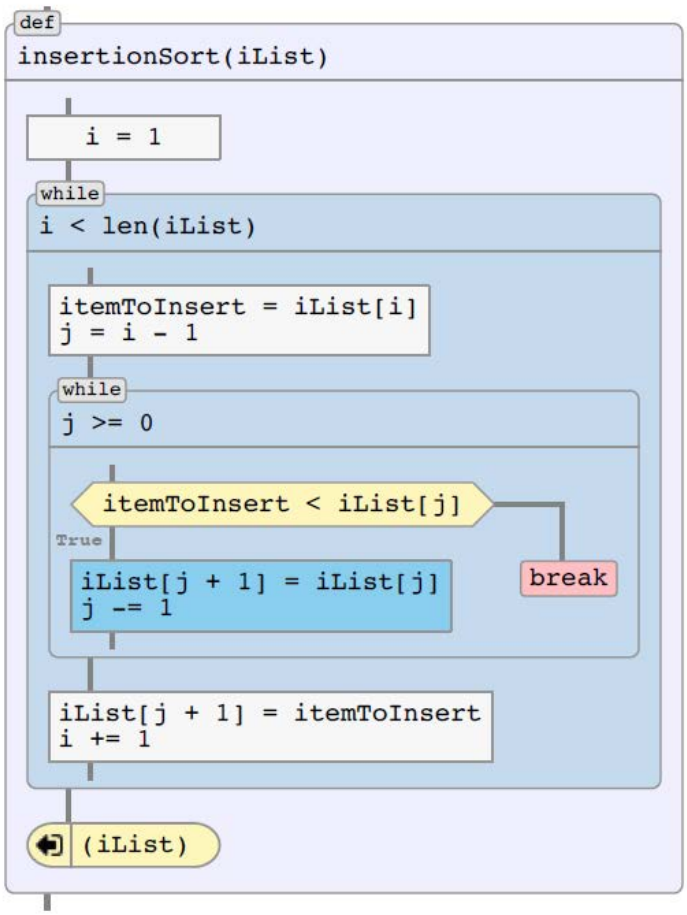
b. Which configuration of items results in the smallest number of swaps in the aforementioned algorithm? Which configuration results in the largest number of swaps?

c. Given the following list of items:

```
[71, 6, 35, 24, 61, 82, 27, 1, 17, 83]
```

Modify the algorithm to display the contents of the list after each iteration of the outer loop when sorting in ascending order.

d. The following design represents a different type of sort algorithms, called `InsertionSort`.



Write the code that implements the `InsertionSort` algorithm, then verify its correctness using the following commands:

```
insertionSort = insertionSort([5, 6, 8, 3, 2, 1, 9, 4])
print(insertionSort)
# The output should be [1, 2, 3, 4, 5, 6, 8, 9]
```

e. Given the list in question **2c**, modify the `InsertionSort` algorithm to display the contents of the list after each iteration of the outer loop when sorting in ascending order.

f. Explain why the `InsertionSort` algorithm performs well on partially sorted lists.

This is the end of lab 7
Tamer Abdou, PhD